Comparison of Classifiers for Handwritten Digit Recognition*

Krzysztof Chrobok[1], Alicja Pradela[1]

[1]Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND

Abstract

This paper presents a comparative analysis of three simple machine learning algorithms (k-Nearest Neighbors, Naive Bayes and Decision Tree) applied to the MNIST handwritten digit dataset. Various parameters for the algorithms are tested: multiple k values, uniform and weighted voting for k-NN, Gaussian, Multinomial and Bernoulli Naive Bayes with PCA reduction for Gaussian NB, various max depths, Gini index and Shannon information gain criterions for Decision Trees. kNN achieves the best accuracy, while PCA + Gaussian Naive Bayes achieves the fastest total runtime with satisfactory accuracy.

Keywords

knn, naive bayes, decision tree, mnist, digit recognition

1. Introduction

Handwriting recognition, often referred to as handwriting Optical Character Recognition (OCR) is a specialized subfield of optical character recognition that transforms handwritten characters into digital text or commands. Within this domain, handwritten digit recognition focuses on identifying human-written numerals from diverse sources such as scanned images, paper forms, and touchscreens, and classifying them into one of ten categories (0-9). [1]

Despite decades of development, handwritten digit recognition remains a significant challenge for computers due to the unavoidable variability and imperfections in human handwriting – some digits are written neatly, while others can be almost illegible even to the human eye. This inconsistency makes research in handwritten text recognition especially important. There are numerous practical applications of digit recognition, including driverless cars, identifying number plates, banking, and more. As a result, research into handwritten digit recognition has intensified in recent years, and as computational power increases, they continue to achieve higher accuracy and robustness. [2] There are also several other models used in shape detection and processing of images like pose detection [3] or multi-stage transformers applicable to complex ideas [4]. Very important for efficiency of the solution are training and development together with data analytics processing. In [5] novel class discovery method was proposed for data analytics to improve model training iv variety of configurations, while [6] proposed dynamic center point learning for difficulties in input data.

In this paper, we present a comparative analysis of three simple machine learning algorithms – k-Nearest Neighbors (k-NN), Naive Bayes, and Decision Trees applied to the MNIST dataset. For k-NN, we vary the number of neighbors and contrast uniform against distance-weighted voting. In the Naive Bayes experiments, we evaluate Gaussian, Multinomial, and Bernoulli variants, applying PCA dimensionality reduction to the Gaussian model to 50 features [7]. Decision Trees are tested across a range of maximum depths using both Gini impurity and Shannon information gain to split nodes. Our objective is to evaluate each of these methods' accuracy and computational performance, examining a variety of parameter settings, thereby providing guidance for selecting the most suitable algorithm for specific application requirements.

*IVUS 2025: Information Society and University Studies 2025, May 15, Kaunas, Lithuania

kc310443@student.polsl.pl (K. Chrobok); ap311009@student.polsl.pl (A. Pradela)

2. Methodology

2.1. k-Nearest Neighbor

k-NN is based on computing a distance metric between the test sample and training samples. Euclidean distance is commonly used as the distance metric, and is defined as such, with x, y being the samples with n total features and x_i, y_i are the indivdual sample features [8]:

$$\mathrm{d}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1}$$

The `scikit-learn` Python library used for our evaluation uses Minkowski distance by default, with order $p = 2$, which is the Euclidean distance [9, 10].

After computing distance metrics between the the test sample and all training samples, k closest samples are selected and based on their classes a prediction is made. If they're all in a single class A, the test sample is also classified as A. If the neighbors belong to various classes, two approaches are possible: simple majority voting or distance weighted voting. In simple majority voting the test sample gets classified into the class with the most neighbors, in distance weighted voting the closer a neighbor is, the more weight it has when classifying the test sample [11]. We compare both of those methods in our analysis.

2.2. Naive Bayes

Naive Bayes classifiers are a family of supervised learning models based on the Bayes' theorem, which computes the posterior probability of each class c given an input characteristic vector $\mathbf{x} = (x_1, x_2, \ldots, x_D)$. In our MNIST dataset, each image undergoes a flattening process to form a $D = 28 \times 28 = 784$ dimensional vector, where $c \in \{0, 1, \ldots, 9\}$ denotes the ten possible digit labels. Under the "naive" assumption that all features are conditionally independent given the class, the likelihood $P(\mathbf{x} \mid c)$ is expressed as:

$$P(\mathbf{x} \mid c) = \prod_{j=1}^{D} P(x_j \mid c). \tag{2}$$

By Bayes' theorem,

$$P(c \mid \mathbf{x}) = \frac{P(c)\, P(\mathbf{x} \mid c)}{P(\mathbf{x})} \;\propto\; P(c) \prod_{j=1}^{D} P(x_j \mid c) \tag{3}$$

and classification reduces to the Maximum A Posteriori (MAP) rule:

$$\hat{c} = \arg\max_{c} P(c) \prod_{j=1}^{D} P(x_j \mid c) \tag{4}$$

Here $P(c)$ is the prior probability of class c, estimated as the fraction of training samples in that class. The variations of naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_j \mid c)$ [12].

2.2.1. Gaussian Naive Bayes

The Gaussian Naive Bayes classifier assumes that features follow a normal distribution, with parameters $\mu_{c,j}$ and $\sigma^2_{c,j}$ representing the mean and variance of feature x_j for class c. The class-conditional likelihood is given by:

$$P(x_j \mid c) = \frac{1}{\sqrt{2\pi\sigma_{c,j}^2}} \exp\left(-\frac{(x_j - \mu_{c,j})^2}{2\sigma_{c,j}^2}\right) \tag{5}$$

2.2.2. Multinomial Naive Bayes

Multinomial Naive Bayes is designed for count-based features. Each class-feature pair uses smoothed maximum likelihood:

$$\theta_{c,j} = \frac{N_{c,j} + \alpha}{\sum_k N_{c,k} + \alpha D} \tag{6}$$

where $N_{c,j}$ is the total count of feature j and α is a smoothing parameter to prevent zero probabilities.. [13, 14].

2.2.3. Bernoulli Naive Bayes

Bernoulli Naive Bayes assumes binary features (e.g., binarized pixels) and models the likelihood by considering both the presence and absence of each feature:

$$P(x_i \mid y) = P(x_i = 1 \mid y) \cdot x_i + (1 - P(x_i = 1 \mid y)) \cdot (1 - x_i) \tag{7}$$

In `scikit-learn`, `BernoulliNB` handles binarization internally [15].

2.3. Decision Trees

Decision Trees are tree-structured classification models learned in a top-down fashion with recursive partirioning [16]. Every step divides the data based on a single feature. Best feature to split on is selected through a variety of criterions such as: Shannon Information Gain, Gini Index, DKM Criterion, Gain Ratio, Kolmogorov-Smirnov Criterion or others [17]. We compare the Shannon Information Gain and Gini Index criterions in our analysis.

When the Shannon Information Gain criterion is used, the following information gain function is maximized [18]:

$$-\sum_{i=1}^{k} p\left(c_i\right) \log_2\left(p\left(c_i\right)\right) + \sum_{i=1}^{n} p\left(t_i\right) \sum_{j=1}^{k} p\left(c_j|t_i\right) \log_2\left(p\left(c_j|t_i\right)\right) \tag{8}$$

When the Gini Index criterion is used, the following function is maximized [18]:

$$1 - \sum_{i=1}^{k} \left(p\left(c_i\right)\right)^2 - \sum_{i=1}^{n} p\left(t_i\right) \sum_{j=1}^{k} p\left(c_j|t_i\right) \left(1 - p\left(c_j|t_i\right)\right) \tag{9}$$

In both equations, the following variables are used:

- c_i – class i out of k existing classes,
- $p\left(c_i\right)$ – probability of an object belonging to class c_i,
- t_i – branch i out of all n branches resulting from a split on a feature,
- $p\left(t_i\right)$ – probability of an object belonging to branch t_i,
- $p\left(c_j|t_i\right)$ – conditional probability of an object belonging to class c_i if it belongs to branch t_i

Table 1
k-Nearest Neighbor benchmark results

k	Prediction voting	Accuracy	Mean time to predict the entire test set
3	Uniform	97.05%	13.49s
5	Uniform	96.88%	13.17s
7	Uniform	96.94%	13.16s
3	Weighted	97.17%	12.51s
5	Weighted	96.91%	12.50s
7	Weighted	97.00%	12.81s

3. Experiments

All algorithms were benchmarked on a machine with AMD Ryzen 5 3600 as the CPU and 32 GB RAM (17.7 GB/s reported by Memtest86+). All code used was written in Python using the `scikit-learn` library, with `n_jobs` set to -1 where possible. No other optimizations were applied.

As the training and test set are constant, the accuracy is always the same. Time might differ depending on CPU load, therefore the benchmarking is done as follows:

- Run classifier 3 times as warm-up
- Run classifier 30 times measuring time taken for each iteration
- Take the mean of measurements

3.1. Dataset

MNIST is a database of size-normalized and centered handwritten digit images of size 28x28 pixels. It contains 70000 grayscale images (60000 training images, 10000 test images), out of which 4 are labeled incorrectly [19, 20].

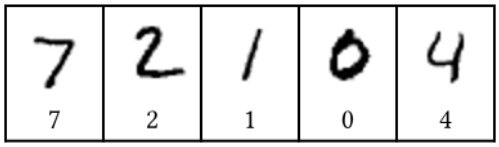

Figure 1: First 5 digits of the MNIST train dataset with labels

3.2. k-NN

We tested the k-Nearest Neighbor algorithm with multiple prime $k \in \{3, 5, 7\}$ values, with both uniform and weighted voting. Benchmark results are included in Table 1.

As expected, we observed negligible differences in speed, under a second for the biggest difference. Because all selected k values achieve similar accuracy, we will consider $k = 5$ (both uniform and weighted) in comparison to other algorithms – the default `scikit-learn` setting [9].

3.3. Naive Bayes

We compared four Naive Bayes variants — GaussianNB, PCA-enhanced GaussianNB, MultinomialNB and BernoulliNB — and presented their accuracy and runtime in Table 2.

Raw `GaussianNB` achieves only 55.4% accuracy because it treats each pixel as an independent normally distributed feature, which doesn't perform well with the complex, multi-modal distributions

Table 2
Naive Bayes benchmark results

Model	Accuracy	Mean training time	Mean total time
GaussianNB	55.4%	0.47 s	1.03 s
PCA + GaussianNB	87.8%	0.51 s	0.56 s
MultinomialNB	83.7%	1.79 s	1.80 s
BernoulliNB	84.3%	2.04 s	2.10 s

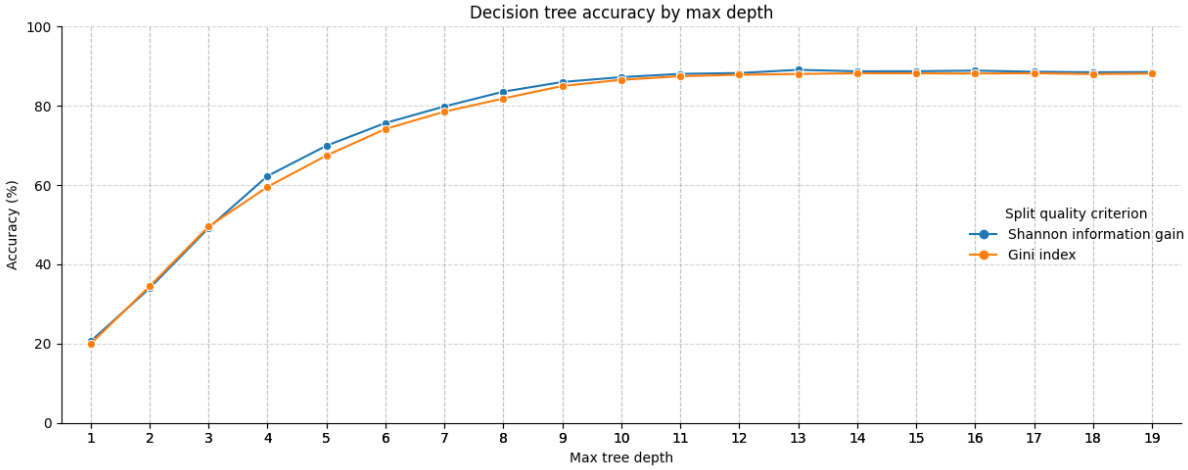

Figure 2: Decision tree accuracy by max depth and split criterion

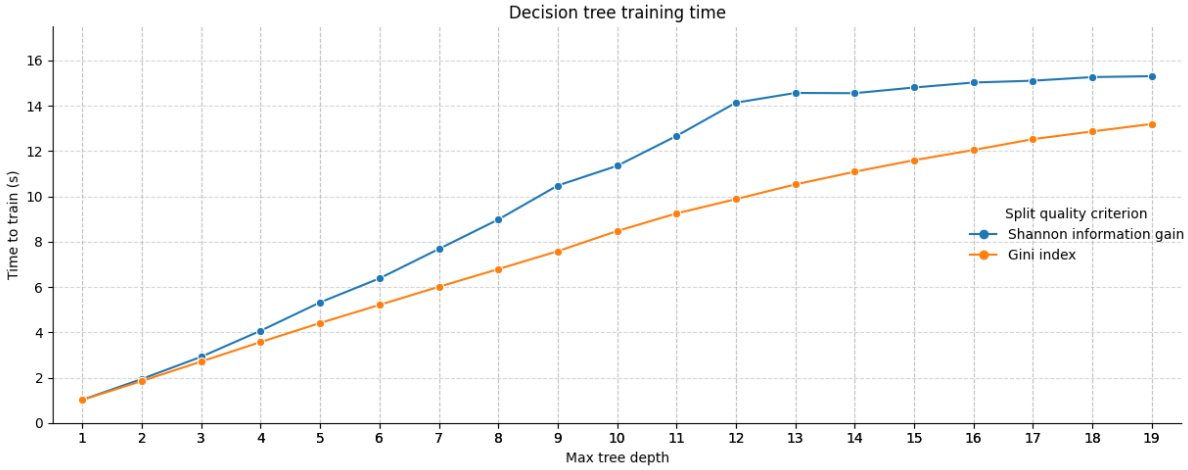

Figure 3: Time to train a decision tree by max depth and split criterion

of handwritten digits. To address this, we applied PCA with 50 components, reducing the data's 784 dimensions by focusing on the directions with the most variance. This helped reduce noise and created a space where `GaussianNB`'s assumption of normality worked better — with accuracy reaching 87.8% while also reducing the prediction runtime [21]. Because of this we will not consider regular `GaussianNB` in comparison to other algorithms, only the PCA-enhanced version.

The discrete-feature classifiers `MultinomialNB`, which treats pixel intensities as counts, and `BernoulliNB`, which encodes pixels as binary values, achieve 83.7% and 84.3% accuracy, respectively; although they require longer training times, their performance approaches that of the PCA-enhanced `GaussianNB`.

Table 3
Decision Tree benchmark results

Tree max depth	Shannon Information Gain		Gini Index	
	Mean time to learn	Accuracy	Mean time to learn	Accuracy
1	1.02s	20.6%	1.02s	19.94%
2	1.94s	33.98%	1.85s	34.47%
3	2.92s	49.18%	2.71s	49.53%
4	4.07s	62.3%	3.57s	59.57%
5	5.32s	69.95%	4.41s	67.47%
6	6.38s	75.68%	5.21s	74.16%
7	7.68s	79.85%	6.01s	78.53%
8	8.98s	83.58%	6.79s	81.86%
9	10.48s	86.02%	7.58s	85.01%
10	11.35s	87.24%	8.47s	86.57%
11	12.67s	88.06%	9.25s	87.48%
12	14.14s	88.28%	9.88s	87.85%
13	14.57s	89.08%	10.53s	88.05%
14	14.56s	88.72%	11.09s	88.22%
15	14.81s	88.73%	11.60s	88.2%
16	15.03s	88.89%	12.05s	88.16%
17	15.11s	88.6%	12.53s	88.22%
18	15.27s	88.48%	12.87s	88.02%
19	15.31s	88.53%	13.20s	88.14%

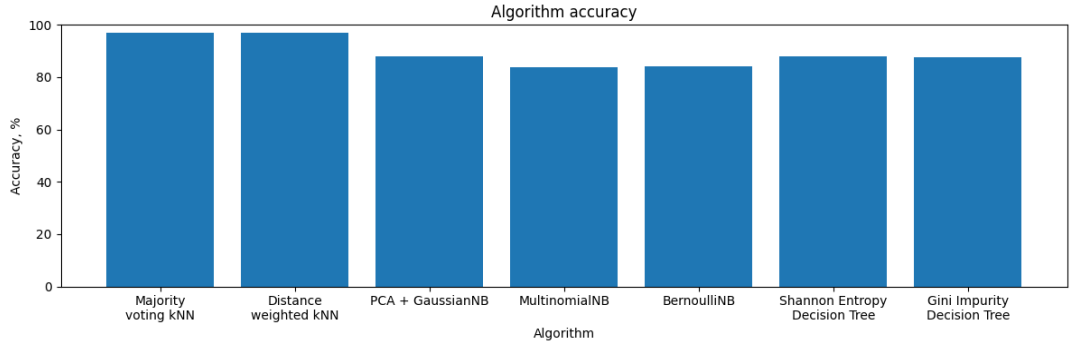

Figure 4: Algorithm accuracy

3.4. Decision Trees

We tested the Decision Tree algorithm with multiple maximum tree depths from 1 to 19, using both Shannon Infromation Gain and Gini Index. Benchmark results are included in Table 3 and on Figures 2, 3. Predicion benchmarks are ommited, as they are negligible (mean 0.02s).

We can observe both criterion achieve similar accuracy, however Gini Index has a consistently lower training time than Shannon Information Gain. We will consider max tree depth 11 in comparison to other algorithms, with both criterions.

4. Conclusion

As we can see on Figure 4, kNN has the best accuracy out of all algorithms tested. It is also the slowest, so if only accuracy is a concern and runtime is not, it should be used. Both distance weighted and majority voting methods achieve similar accuracy results, so either one can be selected. If runtime is a concern then PCA + Gaussian Naive Bayes should be selected – it has the lowest runtime total (training + prediction) while achieving satisfactory accuracy. Both Decision Trees achieve similar accuracy to

PCA-enhanced GaussianNB. They take much longer to train however are the fastest to predict the entire set, so they should be used if prediction time is the biggest concern.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] F. Essam, H. Samy, J. Wagdy, S. Albert, D. S. AbeElminaamb, Mlhandwrittenrecognition: Handwritten digit recognition using machine learning algorithms, Journal of Computing and Communication 2 (2023). URL: https://journals.ekb.eg/article_282076.html. doi:10.21608/jocc.2023.282076, accessed: 2025-05-12.

[2] H. Khandelwal, MNIST Digit Classification using Machine Learning, Technical Report, Jaypee University of Information Technology, Solan, H.P., 2023. URL: http://ir.juit.ac.in:8080/jspui/handle/123456789/9983, supervised by Pardeep Kumar.

[3] Z. Tian, W. Fu, M. Woźniak, S. Liu, Pcdpose: enhancing the lightweight 2d human pose estimation model with pose-enhancing attention and context broadcasting, Pattern Analysis and Applications 28 (2025) 59.

[4] X. Yang, Y. Yang, S. Ma, Z. Li, W. Dong, M. Woźniak, Samt-generator: A second-attention for image captioning based on multi-stage transformer network, Neurocomputing 593 (2024) 127823.

[5] Q. Yan, Y. Yang, Y. Dai, X. Zhang, K. Wiltos, M. Woźniak, W. Dong, Y. Zhang, Clip-guided continual novel class discovery, Knowledge-Based Systems 310 (2025) 112920.

[6] Y. Hu, A. Niu, J. Sun, Y. Zhu, Q. Yan, W. Dong, M. Woźniak, Y. Zhang, Dynamic center point learning for multiple object tracking under severe occlusions, Knowledge-Based Systems 300 (2024) 112130.

[7] A. Popiel, M. Woźniak, Feature reduction for simplified handwritten digit classification model, in: 2022 IEEE Symposium Series on Computational Intelligence (SSCI), 2022, pp. 448–454. doi:10.1109/SSCI51031.2022.10022258.

[8] L. E. Peterson, K-nearest neighbor, Scholarpedia 4 (2009) 1883. doi:10.4249/scholarpedia.1883, revision #137311.

[9] scikit-learn developers, sklearn.neighbors.KNeighborsClassifier — scikit-learn documentation, 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html, accessed: 2025-05-04.

[10] SciPy developers, scipy.spatial.distance.minkowski — SciPy v1.11.0 Manual, 2025. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.minkowski.html, accessed: 2025-05-04.

[11] P. Cunningham, S. J. Delany, k-nearest neighbour classifiers: 2nd edition (with python examples), CoRR abs/2004.04523 (2020). URL: https://arxiv.org/abs/2004.04523. arXiv:2004.04523.

[12] scikit-learn developers, sklearn.naive_bayes — scikit-learn documentation, 2025. URL: https://scikit-learn.org/stable/modules/naive_bayes.html, accessed: 2025-05-11.

[13] Sriram, Multinomial naive bayes explained: Function, advantages & disadvantages, applications, 2025. URL: https://www.upgrad.com/blog/multinomial-naive-bayes-explained/, accessed: 2025-05-11.

[14] scikit-learn developers, sklearn.naive_bayes.MultinomialNB — scikit-learn documentation, 2025. URL: https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes, accessed: 2025-05-11.

[15] scikit-learn developers, sklearn.naive_bayes.BernoulliNB — scikit-learn documentation, 2025. URL: https://scikit-learn.org/stable/modules/naive_bayes.html#bernoulli-naive-bayes, accessed: 2025-05-11.

[16] J. Fürnkranz, Decision Tree, Springer US, Boston, MA, 2016, pp. 1–5. URL: https://doi.org/10.1007/978-1-4899-7502-7_66-1. doi:10.1007/978-1-4899-7502-7_66-1.

[17] L. Rokach, O. Maimon, Decision Trees, Springer US, Boston, MA, 2005, pp. 165–192. URL: https://doi.org/10.1007/0-387-25465-X_9. doi:10.1007/0-387-25465-X_9.

[18] L. E. Raileanu, K. Stoffel, Theoretical comparison between the gini index and information gain criteria, Annals of Mathematics and Artificial Intelligence 41 (2004) 77–93. URL: https://doi.org/10.1023/B:AMAI.0000018580.96245.c6. doi:10.1023/B:AMAI.0000018580.96245.c6.

[19] Y. LeCun, Mnist handwritten digit database, https://web.archive.org/web/20020622183530/http://yann.lecun.com/exdb/mnist/, 2002.

[20] N. M. Müller, K. Markert, Identifying mislabeled instances in classification datasets, in: 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1–8. doi:10.1109/IJCNN.2019.8851920.

[21] scikit-learn developers, sklearn.decomposition.PCA — scikit-learn documentation, 2025. URL: https://scikit-learn.org/stable/modules/decomposition.html#pca, accessed: 2025-05-11.